

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD
OF PATENT APPEALS AND INTERFERENCES**

In re Application of: Ping-Sheng Tseng

Serial No.: 09/954,715

Confirmation No.: 8847

Filed: September 12, 2001

For: COMMON SHARED MEMORY IN A
VERIFICATION SYSTEM

§
§
§
§
§
§
§
§
§

Group Art Unit: 2128

Examiner: Akash Saxena

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

APPEAL BRIEF

Appellants submit this Appeal Brief to the Board of Patent Appeals and Interferences in the above-identified application. The Commissioner is hereby authorized to charge counsel's Deposit Account No. 50-3562 for any fees, including extension of time fees or excess claim fees, required to make this response timely and acceptable to the Office.

Table of Contents

Appeal Brief Section	Page Number
REAL PARTY IN INTEREST	3
RELATED APPEALS AND INTERFERENCES.....	3
STATUS OF CLAIMS.....	3
STATUS OF AMENDMENTS.....	3
SUMMARY OF CLAIMED SUBJECT MATTER	4
GROUND OF OBJECTION AND REJECTION TO BE REVIEWED ON APPEAL.....	9
ARGUMENT.....	10
CONCLUSION	20
CLAIMS APPENDIX.....	21
EVIDENCE APPENDIX.....	33
RELATED PROCEEDINGS APPENDIX	34

REAL PARTY IN INTEREST

The real party in interest is Verisity Design, Inc., which is a subsidiary of Cadence Design Systems, Inc., located in San Jose, California.

RELATED APPEALS AND INTERFERENCES

No other appeals or interferences that are directly affected by, or have a bearing on the Board's decision in the pending Appeal are known to Appellants, Appellants' legal counsel, or the Assignee.

STATUS OF CLAIMS

Claims 1-50 are pending in the application. Claims 1-50 were finally rejected in the Final Office Action, as discussed in detail below. The rejections of claims 1-50 are presently appealed.

STATUS OF AMENDMENTS

No amendments to the claims were submitted in this application subsequent to the Final Office Action.

SUMMARY OF CLAIMED SUBJECT MATTER

Aspects of the invention relate to modeling a circuit design using both software and hardware. The entire circuit design can be modeled in software on a computing system, whereas evaluation components (e.g., register components, combinational components, etc.) can be modeled in hardware using reconfigurable hardware. Since the computing system models the entire design in software, the computing system accesses internal states of the reconfigurable hardware to update the software model.

In particular, Appellants claim 1 recites an electronic design automation system for verifying a user design. The system comprises:

- a computing system that includes a central processing unit for modeling the user design in software (Appellants' specification, p. 22, lines 1-29; workstation 10 in FIG. 1; Appellants' specification, p. 264, line 8 to p. 265, line 7; RCC computing system 2081 in FIG. 67);

- an internal bus system coupled to the computing system (Appellants' specification, p. 20, lines 17-18; PCI bus 50 in FIG. 1; PCI interface 2085 in FIG. 67);

- reconfigurable hardware logic coupled to the internal bus system and for generating a hardware model which includes at least a portion of the user design modeled in hardware (Appellants' specification, p. 23, lines 13-25; reconfigurable hardware model 20 in FIG. 1; Appellants' specification, p. 265, lines 8-18; RCC hardware array 2084 in FIG. 67);

- control logic coupled to the internal bus system for controlling the delivery of data between the reconfigurable hardware logic and the computing system (Appellants' specification, p. 20, lines 3-12; p. 272, lines 2-12; p. 278, lines 11-19; FIG. 69, elements 2152, 2156, 2158, 2159, 2165; FIGs. 70-73); and

- shared memory for holding a first information of a software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model (Appellants' specification, p. 52, lines 21-26; p. 55, lines 8-16; memory 15 in FIG. 1).

Appellants' claim 19 recites a method of a method of simulating a circuit in a simulation system, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections. The method comprises:

- determining component type in the hardware language (Appellants'

specification, p. 30, lines 15-21; step 302 in FIG. 4);

generating a software model of the circuit (Appellants' specification, p. 32, lines 16-19; step 304 in FIG. 4);

generating a hardware model of at least a portion of the circuit based on component type automatically (Appellants' specification, p. 33, lines 8-21; step 307 in FIG. 4);

allocating space in a shared memory for the software model and the hardware model (Appellants' specification, p. 33, line 22 through p. 34, line 3; step 308 in FIG. 4; Appellants' specification, p. 55, lines 8-16);

storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model (Appellants' specification, p. 55, lines 8-16; REG, S2H, H2S, CLK buffers in FIG. 10); and

simulating the behavior of the circuit with the software model by initially using the first information in the shared memory (Appellants' specification, p. 34, line 24 through p. 36, line 13; FIG. 5).

Appellants' claim 24 recites a method of simulating a circuit, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections. The method comprises:

generating a software model of the circuit (Appellants' specification, p. 32, lines 16-19; step 304 in FIG. 4);

generating a hardware model of the circuit (Appellants' specification, p. 33, lines 8-21; step 307 in FIG. 4);

allocating space in a shared memory for the software model and the hardware model (Appellants' specification, p. 33, line 22 through p. 34, line 3; step 308 in FIG. 4; Appellants' specification, p. 55, lines 8-16);

storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model (Appellants' specification, p. 55, lines 8-16; REG, S2H, H2S, CLK buffers in FIG. 10);

simulating a behavior of the circuit with the software model by providing

input data to the software model (Appellants' specification, p. 34, line 24 through p. 36, line 13; FIG. 5);

selectively switching to the hardware model through software control (Appellants' specification, p. 35, lines 15-29; steps 334-336 in FIG. 5);

providing input data to the hardware model (Appellants' specification, p. 56, lines 1-7; p. 35, lines 15-13; step 335 in FIG. 5); and

evaluating the input data in the hardware model based on the detection of a trigger event in the software model (Appellants' specification, p. 35, lines 24-29; step 334 in FIG. 5; p. 56, lines 11-19).

Appellants' claim 30 recites a method of evaluating data in a circuit during a simulation process. The method comprises:

generating a software model of the circuit (Appellants' specification, p. 32, lines 16-19; step 304 in FIG. 4);

generating a hardware model of at least a portion of the circuit (Appellants' specification, p. 33, lines 8-21; step 307 in FIG. 4);

allocating space in shared memory for the software model and the hardware model (Appellants' specification, p. 33, line 22 through p. 34, line 3; step 308 in FIG. 4; Appellants' specification, p. 55, lines 8-16);

storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model (Appellants' specification, p. 55, lines 8-16; REG, S2H, H2S, CLK buffers in FIG. 10);

propagating data to the hardware model until the data stabilizes (Appellants' specification, p. 56, lines 1-7; p. 35, lines 15-13; step 335 in FIG. 5);

detecting a clock edge in the software model (Appellants' specification, p. 35, lines 15-13; step 334 in FIG. 5); and

evaluating data with the hardware model in response to the clock edge detection in the software model and in synchronization with a hardware-generated clock (Appellants' specification, Appellants' specification, p. 35, lines 24-29; step 334 in FIG. 5; p. 56, lines 11-19; p. 91, lines 3-16).

Appellants' claim 33 recites a simulation system operating in a host computer system for simulating a behavior of a circuit, the host computer system including a central

processing unit (CPU), shared memory, and a local bus coupling the CPU to main memory and allowing communication between the CPU and main memory, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections. The system comprises:

- a software model of the circuit coupled to the local bus (Appellants' specification, p. 55, lines 8-23; elements 315 and 328 in FIG. 10);

- software control logic coupled to the software model (simulator kernel 316 in FIG. 10) and a hardware logic element (element 325 in FIG. 10), for controlling the operation of the software model and said hardware logic element (Appellants' specification, p. 55, lines 9-16);

- said hardware logic element coupled to the local bus and including a hardware model of at least a portion of the circuit configured automatically based on component type (Appellants' specification, p. 55, lines 17-29; Appellants' specification, p. 33, lines 8-21; FIG. 10)

- DMA engine for loading state information of the hardware model, where the state information comprises at least one internal state of the hardware model, from the hardware logic element to the shared memory (Appellants' specification, p. 56, lines 1-7; elements 321-324 in FIG. 10); and

- the software model is capable of directly accessing the state information of the hardware model (Appellants' specification, p. 55, lines 8-16; REG, S2H, H2S, CLK buffers in FIG. 10).

Appellants' claim 38 recites a coverification system for verifying a user design.

The system comprises:

- a computing system including a central processing unit and memory for modeling the user design in software (Appellants' specification, p. 22, lines 1-29; workstation 10 in FIG. 1; Appellants' specification, p. 264, line 8 to p. 265, line 7; RCC computing system 2081 in FIG. 67);

- an internal bus system coupled to the computing system (Appellants' specification, p. 20, lines 17-18; PCI bus 50 in FIG. 1; PCI interface 2085 in FIG. 67);

- reconfigurable hardware logic coupled to the internal bus system and for modeling at least a portion of the user design in hardware (Appellants' specification, p. 23, lines 13-25; reconfigurable hardware model 20 in FIG. 1; Appellants' specification, p. 265, lines 8-18; RCC hardware array 2084 in FIG. 67);

- an external interface coupled to the internal bus system and at least one external device (Appellants' specification, p. 270, lines 12-29; external interface

2139 and target system 2120 in FIG. 69);

control logic coupled to the internal bus system for controlling the delivery of data among the reconfigurable hardware logic, the computing system, and the external interface (Appellants' specification, p. 20, lines 3-12; p. 272, lines 2-12; p. 278, lines 11-19; FIG. 69, elements 2152, 2156, 2158, 2159, 2165; FIGs. 70-73); and

shared memory for loading state information of the user design, where the state information comprises at least one internal state of the user design, from the reconfigurable hardware logic to the computing system (Appellants' specification, p. 52, lines 21-26; p. 55, lines 8-16; memory 15 in FIG. 1).

Appellants' claim 44 recites a method of verifying the proper operation of a user design connected to an external I/O device. The method comprises:

generating a first model of the user design in software (Appellants' specification, p. 32, lines 16-19; step 304 in FIG. 4);

loading the first model in a shared memory (Appellants' specification, p. 29, lines 12-18; FIG. 3);

generating a second model of at least a portion of the user design in hardware (Appellants' specification, p. 33, lines 8-21; step 307 in FIG. 4);

controlling the second model in hardware with the first model in the software (Appellants' specification, p. 35, lines 15-29; steps 334-336 in FIG. 5); and

loading state information from the second model, where the state information comprises at least one internal state of the second model, to the shared memory and the first model is capable of directly accessing the state information of the second model (Appellants' specification, p. 55, lines 8-16; REG, S2H, H2S, CLK buffers in FIG. 10).

GROUND OF OBJECTION AND REJECTION TO BE REVIEWED ON APPEAL

I. Rejection of claims 1-6, 19-36, 38, and 44-46 under 35 U.S.C. §103(a) as being unpatentable over Bhandari et al. (U.S. Patent No. 5,663,900) ("BH '900") in view of Klein (U.S. Patent No. 5,771,370) ("Klein").

II. Rejection of claims 7 and 10 under 35 U.S.C. §103(a) as being unpatentable over BH '900 and Klein in further view of IEEE article "Q-Modules: Internally Clocked Delay-Insensitive Modules" by Fred U. Rosenberger ("RO '1998").

III. Rejection of claims 8-9 and 11-18 under 35 U.S.C. §103(a) as being unpatentable over BH '900, Klein, and RO '1998 in further in view IEEE article "High Speed External Asynchronous/Internally Clocked Systems" by W.S. VanSheik et al. ("VA '1997").

IV. Rejection of claim 37 under 35 U.S.C. §103(a) as being unpatentable over BH '900 and Klein in further in view of U.S. Patent No. 5,661,662 issued August 26, 1997 to Butts et al. ("Butts").

V. Rejection of claims 39-43 and 47-50 under 35 U.S.C. §103(a) as being unpatentable over BH '900 and Klein in further view of "A Heterogeneous Environment for Hardware/Software Cosimulation" by William D. Bishop et al. (IEEE Transactions on Computers, 1997, pp. 14-22) ("BI '1997"). .

ARGUMENT

I. THE EXAMINER ERRED IN REJECTING CLAIMS 1-6, 19-36, 38, and 44-46 UNDER 35 U.S.C. §103(a) BECAUSE THE COMBINATION OF BH '900 AND KLEIN FAILS TO TEACH, SUGGEST OR OTHERWISE RENDER OBVIOUS THE CLAIMS.

The Examiner stated that BH '900 teaches an EDA system for verification and modeling of a user design having a computer system, and internal bus, and reconfigurable hardware logic. (Final Office Action, p. 7). The Examiner further stated that BH '900 teaches "that the second information comprises at least one internal state of the hardware model," taking Official Notice that such "information would be vital to the [sic] any hardware-software co-simulation system." (Final Office Action, p. 8) (citing U.S. Patent 6,052,524 and U.S. Patent 5,546,562 as evidentiary support for the Official Notice). The Examiner conceded that BH '900 does not disclose a shared memory, as recited in Appellants' claims. (Final Office Action, p. 8).

That Examiner stated, however, that BH '900 does disclose communication between the software model and the hardware model and signal processing there-between. The Examiner argues that this teaching renders obvious Appellants' shared memory. (Final Office Action, p. 8). Furthermore, the Examiner cited Klein as teaching a shared, coherent memory for holding first information of the software model and second information of the hardware model, where the software model is capable of directly accessing the second information. (Final Office Action, pp. 8-9). The Examiner concluded that it would have been obvious to use the coherent memory of Klein in the system of BH '900, stating that Klein "enhances the performance of [the] BH '900 hardware-software model by providing the shared memory." (Final Office Action, p. 9).

BH '900 teaches an electronic simulation and emulation system for simulating or emulating a functional specification of a prototype design. (See BH '900, Abstract). In particular, BH '900 teaches a software simulator having various simulation models that define the prototype design. Interface software and hardware are provided between the software simulator and external systems. (BH '900, col. 3, lines 6-29). The external systems may include a functional tester, logic analyzer, emulator, modeler, etc. The

external systems cooperate functionally with, or monitor signals from, the software simulator by transferring data signals there-between. (BH '900, col. 3, lines 55-60).

First, Appellants traverse the Examiner's official notice. It is not common knowledge well-known in the art that information comprising at least one internal state of the hardware model is vital to any hardware-software co-simulation system. In support of the official notice, the Examiner cited U.S. patent 6,052,524, col. 14, lines 21-37, which states:

Co-verification simulators known in art typically determine the internal behavior and state of hardware and software components every cycle of a simulator. This provides accuracy, but may slow the simulation by an order of magnitude or more. Cycle-accurate simulator 12 in a preferred embodiment of the present invention allows events to be set and the internal behavior and state of hardware components to be accurately examined at specific events (e.g., after a memory access). This allows cycle-accurate simulator 12 to provide faster simulations of hardware and software components. Cycle-accurate simulate 12 can also determine internal behavior and state of hardware and software components every cycle by setting a first event to occur on a first simulator cycle and a second event to occur on a second simulator cycle, a third event to occur on a third simulator cycle, etc.

Nothing in the cited passage, however, indicates that information comprising at least one internal state of the hardware model is "vital" to a co-simulation system. The passage states that co-verification systems "typically" determine internal behavior and state of hardware and software components. "Typically" does not mean vital or necessary, but rather indicates that in some cases co-verification systems do not determine internal behavior and state of hardware and software components. Moreover, "determining" internal behavior and state of hardware and software components does not teach or suggest storing them in a shared memory, as recited in Applicants' claim 1.

The Examiner also cited the Abstract of U.S. patent 5,546,562, which states:

An emulation modeling apparatus (54) comprises a combination of a device under simulation (48) to be emulated and means for keeping the device under simulation (48) in a quiescent state at normal operating speeds and in a normal operating sequence so as to allow dual access to the emulation modeling apparatus (54) without loss of data or accuracy of functions. One access is from a host simulation environment (26) while the other is from a model debug user interface (20) where internal architecturally visible registers and status are available to the user for greater debug control on the simulated subsystem within simulation environment (26). Specifically, any of a wide variety of physical VLSI circuits (48) to be modeled is kept in a quiescent state after power-on by a device control (50). It is then

accessed through simulation means by simulated subsystem within a simulation environment (26), to change the architecturally visible internal state of the VLSI circuit (48). Control (50) brings VLSI circuit (48) out of the quiescent state and submits the requested simulated access. After taking the response, control (50) returns VLSI circuit (48) again to its quiescent state so as to keep its internal state current. The response is sent back to simulation environment (26) to update the simulated subsystem. Independently, any user request for accessing the architecturally visible internal state of the circuit is gathered by model debug and user interface (20). Interface (20) enables control (50) to bring VLSI circuit (48) out of the quiescent state and to submit the user request access. Subsequently, control (50) monitors the response and returns VLSI circuit (48) to its quiescent state so as to maintain the internal state of VLSI circuit (48) current. Control (50) then sends the response back to user interface (20). VLSI circuit (48) thus is always kept ready and current for the next request, either from simulation environment (26) or from user interface (20) without having to reset it. If any user defined breakpoint condition is met during the simulated accesses on the VLSI circuit (48), this information is forwarded by control (50) to simulation environment (26) for stopping the simulation and to user interface (20) to update the debug screen accordingly.

Again, nothing in the cited passage indicates that information comprising at least one internal state of the hardware model is "vital" to a co-simulation system. The cited passage describes a particular invention where internal registers and status of a device are available to a user to provide debug control. Nothing in this passage indicates that such is necessary or vital for all co-simulation systems.

The Examiner further stated that "[i]t is well known in the art that co-simulation of hardware and software share information, otherwise it would not be called co-simulation." (Final Office Action, p. 3). Appellants, however, do not claim a general exchange of information between hardware and software. Rather, Appellants recite that the second information comprises at least one internal state of the hardware model. In essence, Appellants claim a shared memory for storing first information of a software model and the second information, which allows the software model to directly access at least one internal state of the hardware model. The mere fact that co-simulation generally involves sharing information between hardware and software does not teach, suggest, or otherwise render obvious the more specific feature of a shared memory configured to store both information of the software model and internal state information of the hardware model, recited in Applicants' claims.

Accordingly, Appellants contend that the Examiner has not provided sufficient evidentiary support for the official notice and that BH '900 does not teach or suggest that the second information comprises at least one internal state of the hardware model.

Second, the Examiner has not set forth a prima facie case of obviousness with respect to BH '900 alone. The Examiner stated "[t]o emphasize that the limitation relating to shared resource (memory specifically), and direct access of the software model to the hardware model, although obvious in BH '900...." This is a conclusory statement and does not set forth any rationale for why access between a software model and a hardware model teaches, suggests, or otherwise renders obvious a shared memory, as recited in Appellants' claims.

Third, Klein discloses co-simulation of a hardware-software system design. (See Klein, Abstract). That is, a hardware portion of a design is simulated along with a software portion of the design. The software portion of the design is simulated using an instruction set simulator (ISS). The hardware portion of the design is simulated using a logic simulator in conjunction with software-based processor and memory models. (Klein, col. 5, lines 6-65; FIG. 2). Both the logic simulator and the ISS are software-based simulators. (Klein, col. 5, lines 43-50; col. 2, lines 6-8). Klein states that "it should be noted that the present invention may be practiced in one or more general or special purpose computer systems." (Klein, col. 6, lines 38-40).

Thus, Klein and Appellants' invention are directed to two different types of simulation. In Klein, a circuit design comprising both hardware and software portions (sometimes referred to as an embedded system – see Klein, col.1, lines 34-65) is simulated using a computer system. In Appellants' invention, a circuit design is simulated using a computer system coupled to a reconfigurable hardware system (e.g., FPGAs). "Co-simulation" in Klein refers to the simultaneous simulation of hardware and software portions of a circuit design on a computer system. In the context of Appellants' invention, "co-simulation" means simulation of a circuit design (whether an embedded system or not) using both a computer and reconfigurable hardware. Klein is completely devoid of any teaching or suggestion of a reconfigurable hardware system for simulation, emulation, hardware acceleration, etc. of a hardware model.

Thus, Klein does not teach or suggest any hardware component (i.e., a shared memory) that can be used in the hardware portion of BH'900 to emulate a portion of a design. Accordingly, Klein does not bridge the substantial gap between BH '900 and Appellants' invention. Namely, Klein does not provide a teaching or suggestion of a shared memory that can be incorporated into or otherwise modify the system of BH '900. Therefore, Appellants contend that no conceivable combination of BH '900 and Klein renders obvious Appellants' invention.

Fourth, the software described in Klein performs the hardware and software simulations with a single coherent view of the memory of the hardware-software system being co-simulated. (Klein, col. 4, lines 15-21). This "single coherent view of memory" in Klein is not a shared memory, as recited in Appellants' claims. Appellants clearly claim a physical shared memory, as the shared memory is included in a system having a computing system, an internal bus, reconfigurable hardware logic, and control logic. One skilled in the art would clearly understand Appellants' shared memory to be physical in the context of the claims and in light of Appellants' specification. Appellants contend that to interpret the claims otherwise is unreasonable. Appellants are not required to preface each element in a claim with the term "physical."

In contrast to Appellants' invention, the "single coherent view of memory" in Klein is an abstract or logical construct of a memory portion of the hardware-software design being simulated. The "memory" referred to in Klein is not the memory of the co-simulation system, but is rather the memory portion of the design being co-simulated. The Examiner's cite to FIG. 1 of Klein actually confirms this teaching. (Final Office Action, p. 5). FIG. 1 of Klein recites "memory of hardware-software system being co-simulated." The plain meaning of that statement is that the memory is part of the hardware-software system, and that such hardware-software system is being co-simulated. There is no contrary meaning in Klein. This logical memory construct described in Klein does not add anything to the communication of physical signals between the simulator and the external systems in BH '900.

The Examiner stated that "[it] is unclear how [a] 'single coherent view of memory' is patentably different than 'physical shared memory.'" (Final Office Action, p. 4). Performing a simulation with a "view" of memory is an abstract concept. A single

coherent view of the memory is not a literal concept, i.e., Klein does not mean that one looks at memory in the ocular sense. The view of memory in Klein is not something tangible that can be held in one's hands. Rather, the single coherent view of memory in Klein refers to software simulation that accounts for all memory in the design being simulated. (Klein, col. 4, lines 15-21). A physical memory, however, is something tangible and is not abstract. A logical software construct does not teach or suggest a physical memory.

The Examiner also stated that Appellants claim a computer system and it is well known in the art that memory present in a computer system is shared by various applications and peripherals. (Final Office Action, p. 4). Appellants do not disagree. However, the general teaching of memory in a computer shared among peripherals and applications does not teach, suggest, or otherwise render obvious the specific feature of a shared memory configured to store first information of a software model and at least one internal state of a hardware model generated in reconfigurable hardware logic, as recited in Appellants' claims. The Examiner must analyze the claims as a whole, including all of the features recited therein. Appellants' claims are much more specific than a computer having a shared memory.

The Examiner further noted that Appellants' claims do not make the distinction between memory in a co-simulation system and memory of a design being co-simulated. (Final Office Action, p. 5). Such an interpretation of Appellants' claims is unreasonable. Appellants clearly claim a system for co-simulating a design. The claims recite a computing system, an internal bus, a reconfigurable hardware logic, a control logic, and a shared memory. No permissible interpretation of Appellants' claims, as understood by one skilled in the art, renders the shared memory part of the design being co-simulated. The shared memory in Appellants' claims is not part of the software model or the hardware model. Rather, the electronic design automation system comprises the shared memory.

In summary, Appellants primarily argue the following:

- 1) The Examiner has not provided sufficient evidentiary support for the official notice and that BH '900 does not teach or suggest that the second information comprises

at least one internal state of the hardware model. The evidentiary support references are deficient;

2) The Examiner has not set forth a prima facie case of obviousness with respect to BH '900 alone, as the Examiner did not set forth any rationale for why access between a software model and a hardware model teaches, suggests, or otherwise renders obvious a shared memory, as recited in Appellants' claims.

3) Klein does not provide a teaching or suggestion of a shared memory that can be incorporated into or otherwise modify the system of BH '900 and thus no conceivable combination of BH '900 and Klein renders obvious Appellants' invention.

4) The "single coherent view of memory" in Klein is not a shared memory, as recited in Appellants' claims. The "memory" referred to in Klein is not the memory of the co-simulation system, but is rather the memory portion of the design being co-simulated.

In view of the foregoing, Appellants contend that the combination of BH '900 and Klein does not teach, suggest, or otherwise render obvious Appellants' invention recited in claims 1, 19, 24, 30, 33, 38, and 44. Each of claims 1, 19, 24, 30, 33, 38, and 44 recite in some fashion a shared memory that stores at least one internal state of the hardware model, such internal state being accessible by the software model. As discussed above, the cited combination does not teach or suggest such features. Claims 2-6, 20-23, 25-29, 31-32, 34-37, and 45-46 depend from claims 1, 19, 24, 30, 33, 38, and 44 and recite additional features thereof. Since the cited combination does not render obvious Appellants' claims 1, 19, 24, 30, 33, 38, and 44, the cited combination also fails to render obvious dependent claims 2-6, 20-23, 25-29, 31-32, 34-37, and 45-46.

II. THE EXAMINER ERRED IN REJECTING CLAIMS 7 AND 10 UNDER 35 U.S.C. §103(a) BECAUSE THE COMBINATION OF BH '900, KLEIN, AND RO '1998 FAILS TO TEACH, SUGGEST OR OTHERWISE RENDER OBVIOUS THE CLAIMS.

Appellants incorporate the arguments presented above with respect to BH '900 and Klein into the instant section. Appellants submit that BH '900 or Klein, alone or in combination, do not render dependent claims 7 and 10 obvious, at least for their

dependency upon independent claim 1 (see above). The addition of RO'1998 does not correct the shortcomings of BH '900 and/or Klein.

Notably, RO'1988 teaches Q-modules without any teaching or suggestion of "shared memory for storing information from both a hardware model and a software shared memory for holding a first information of a software model and a second information of the hardware model, where the software model is capable of directly accessing the second information of the hardware model." Since BH '900, Klein, and/or RO'1998 do not teach or suggest these features, no conceivable combination of these references can teach or suggest Appellants' invention of claim 1. Therefore, dependent claims 7 and 10, which depend either directly or indirectly from claim 1, are nonobvious in view of the cited references and fully satisfy the requirements of 35 U.S.C. §103.

III. THE EXAMINER ERRED IN REJECTING CLAIMS 8-9 and 11-18 UNDER 35 U.S.C. §103(a) BECAUSE THE COMBINATION OF BH '900, KLEIN, RO '1998, AND VA '1997 FAILS TO TEACH, SUGGEST OR OTHERWISE RENDER OBVIOUS THE CLAIMS.

Appellants incorporate the arguments presented above with respect to BH '900, Klein, and of RO'1998 into the instant section. Appellants submit that BH '900, Klein, and/or RO'1998, alone or in combination, do not render obvious dependent claims 8, 9, and 11-18, at least for their dependency upon independent claim 1 (see above). The addition of VA'1997 does not correct the shortcomings of BH '900, Klein, and/or RO'1998.

Notably, VA'1997 teaches delay insensitive modules without any teaching or suggestion of "shared memory for storing information from both a hardware model and a software shared memory for holding a first information of a software model and a second information of the hardware model, where the software model is capable of directly accessing the second information of the hardware model." Since BH '900, Klein, RO'1998, and/or VA'1997, do not teach or suggest these features, no conceivable combination of these references can teach or suggest Appellants' invention of claim 1. Therefore, dependent claims 8, 9, and 11-18, which depend either directly or indirectly from claim 1, are nonobvious in view of the cited references and fully satisfy the requirements of 35 U.S.C. §103.

IV. THE EXAMINER ERRED IN REJECTING CLAIM 37 UNDER 35 U.S.C. §103(a) BECAUSE THE COMBINATION OF BH '900, KLEIN, AND BUTTS FAILS TO TEACH, SUGGEST OR OTHERWISE RENDER OBVIOUS THE CLAIM.

Appellants incorporate the arguments presented above with respect to BH '900, and Klein into the instant section. Appellants submit that BH '900 or Klein, alone or in combination do not render obvious dependent claim 37, at least for its dependency upon independent claim 33 (see above). The addition of Butts does not correct the shortcomings of BH '900 and/or Klein.

Notably, Butts teaches interconnected FPGAs without any teaching or suggestion of "shared memory for storing information from both a hardware model and a software shared memory for holding a first information of a software model and a second information of the hardware model, where the software model is capable of directly accessing the second information of the hardware model." Since BH '900, Klein, and/or Butts, do not teach or suggest these features, no conceivable combination of these references can teach or suggest Appellants' invention of claim 33. Therefore, dependent claim 37, which depends from claim 33, is nonobvious in view of the cited references and fully satisfy the requirements of 35 U.S.C. §103.

IV. THE EXAMINER ERRED IN REJECTING CLAIMS 39-43 AND 47-50 UNDER 35 U.S.C. §103(a) BECAUSE THE COMBINATION OF BH '900, KLEIN, AND BI '1997 FAILS TO TEACH, SUGGEST OR OTHERWISE RENDER OBVIOUS THE CLAIMS.

Appellants incorporate the arguments presented above with respect to BH '900, and Klein into the instant section. Appellants submit that BH '900 or Klein, alone or in combination do not render obvious dependent claims 39-43 and 47-50, at least for their dependency upon independent claims 38 and 44 (see above). The addition of BI'1997 does not correct the shortcomings of BH '900 and/or Klein.

Notably, BI'1997 teaches hardware/software co-simulation without any teaching or suggestion of "shared memory for storing information from both a hardware model and a software shared memory for holding a first information of a software model and a second

information of the hardware model, where the software model is capable of directly accessing the second information of the hardware model.” Since BH ‘900, Klein, and/or BI’1997, do not teach or suggest these features, no conceivable combination of these references can teach or suggest Appellants’ invention of claims 38 and 44. Therefore, dependent claims 39-43 and 47-50, which depend from claims 38 and 44, are nonobvious in view of the cited references and fully satisfy the requirements of 35 U.S.C. §103.

CONCLUSION

For the reasons advanced above, Appellants respectfully urge that the rejections and objections of claims 1-50 are improper. Reversal of the rejections and objections in this appeal is respectfully requested.

Respectfully submitted,

12/16/2007
Date

 /Robert M. Brush/
Robert M. Brush
Registration No. 45,710

MOSER IP LAW GROUP
1030 Broad Street, 2nd Floor
Shrewsbury, NJ 07702

CLAIMS APPENDIX

1. (Previously Presented) An electronic design automation system for verifying a user design, comprising:

- a computing system including a central processing unit for modeling the user design in software;

- an internal bus system coupled to the computing system;

- reconfigurable hardware logic coupled to the internal bus system and for generating a hardware model which includes at least a portion of the user design modeled in hardware;

- control logic coupled to the internal bus system for controlling the delivery of data between the reconfigurable hardware logic and the computing system; and

- shared memory for holding a first information of a software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model.

2. (Original) The system of claim 1, wherein the first information of the software model includes functional information of the user design.

3. (Original) The system of claim 1, wherein the second information of the hardware model includes functional information of the user design.

4. (Previously Presented) The system of claim 1, wherein the second information of the hardware model includes state information of the user design in the reconfigurable hardware logic.

5. (Original) The system of claim 1, wherein the control logic includes a direct memory access (DMA) engine for loading the second information of the hardware model to the shared memory.

6. (Original) The system of claim 5, wherein the second information of the hardware model includes state information of the user design in the reconfigurable hardware logic.

7. (Previously Presented) The system of claim 1, further comprising:

a timing logic associated with at least one latch so that a desired result is obtained regardless of the order of arrival of a data input and a clock signal to the at least one latch.

8. (Previously Presented) The system of claim 7, wherein the timing logic further comprises:

a first logic having a first input for receiving a first data at a first value, a second input, a first output, and a control input for receiving a clock signal; and

a second logic for storing a current value having a first trigger input, a second logic input coupled to the first output, and a second logic output coupled to the second input of the first logic, wherein the second logic updates to the first value of the first data and presents the first data to the second input of the first logic when a trigger signal is received at the first trigger input, regardless of the order of arrival of the clock signal at the control input or the first input to the first logic.

9. (Previously Presented) The system of claim 8, further comprising:

a third logic for storing a new value of the first data and having a third logic input configured to receive the new value, a second trigger input, and a third logic output, where the third logic output is coupled to the first input of the first logic; and

an edge detector having a clock input, a third trigger input, and an edge detector output, where the edge detector output is coupled to the control input, wherein the trigger signal is applied to the second and third trigger inputs at selected times to update the logic apparatus.

10. (Previously Presented) The system of claim 1, further comprising:

a timing logic associated with at least one flip-flop so that a desired result is obtained regardless of the order of arrival of a data input and a clock signal to the at least

one flip-flop.

11. (Previously Presented) The system of claim 10, wherein the timing logic further comprises:

- an input logic for receiving a new input value and a trigger signal;
- a storage logic for storing an old input value and the trigger signal;
- a selection logic coupled to the input logic for receiving the new input value and coupled to the storage logic for receiving the old input value and selecting from one of the new input value and the old input value to generate a flip-flop output; and
- an edge detecting logic for detecting an edge of a clock signal and receiving the trigger signal, wherein the selection logic outputs the new input value upon the reception of the trigger signal.

12. (Original) The flip-flop of claim 11, wherein the storage logic receives the new input value and stores the new input value upon the reception of the trigger signal.

13. (Original) The flip-flop of claim 11, wherein the edge detecting logic is a positive-edge detecting logic for detecting a positive edge of the clock signal.

14. (Previously Presented) The flip-flop of claim 13, wherein the edge detecting logic is coupled to the selection logic and generates a selector signal to the selection logic for selecting either the new input value or the old input value.

15. (Original) The flip-flop of claim 11, wherein the input logic is a D flip-flop which receives the new input value and having a clock input for receiving the trigger signal.

16. (Previously Presented) The flip-flop of claim 11, wherein the storage logic is a D flip-flop whose input is coupled to the flip-flop output of the selection logic and having a clock input for receiving the trigger signal.

17. (Previously Presented) The flip-flop of claim 11, wherein the selection logic further

includes:

- a multiplexer for receiving the new input value and the old input value, and having a selector input for receiving a selector signal from the edge detecting logic, and a mux output,

- an OR gate for receiving the mux output and a set input, and having an OR gate output, and

- an AND gate for receiving the OR gate output and a reset input, and providing the flip-flop output.

18. (Original) The flip-flop of claim 11, wherein the edge detecting logic includes:

- a D flip-flop for receiving a clock signal and a clock input for receiving the trigger signal, and having a D flip-flop output,

- an AND gate for receiving the clock signal and the D flip-flop output, and having a selector output for providing a selector signal.

19. (Previously Presented) A method of simulating a circuit in a simulation system, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising steps:

- determining component type in the hardware language;

- generating a software model of the circuit;

- generating a hardware model of at least a portion of the circuit based on component type automatically;

- allocating space in a shared memory for the software model and the hardware model;

- storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model; and

- simulating the behavior of the circuit with the software model by initially using the first information in the shared memory.

20. (Previously Presented) The method of claim 19, further comprising step:
loading a third information of the hardware model, where the third information comprises at least one of an internal state of the hardware model and an external state of the hardware model, via direct memory access (DMA).
21. (Original) The method of claim 19, further comprising step:
controlling the software model and the hardware model with a software kernel.
22. (Original) The method of claim 21, wherein the step of controlling further comprises steps: determining the presence of input data to the simulation system;
evaluating clock components;
propagating input data to the hardware model;
detecting active clock edge of the clock components in the software model; and
evaluating the input data with the hardware model in response to the active clock edge detection.
23. (Original) The method of claim 19, wherein the step of simulating further comprises:
simulating the behavior of the circuit with the software model for a time period; and
simulating the behavior of the circuit with the hardware model for another time period to accelerate the simulation process.
24. (Previously Presented) A method of simulating a circuit, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising steps:
generating a software model of the circuit;
generating a hardware model of the circuit;
allocating space in a shared memory for the software model and the hardware model;
storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at

least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model;

simulating a behavior of the circuit with the software model by providing input data to the software model;

selectively switching to the hardware model through software control;

providing input data to the hardware model; and

evaluating the input data in the hardware model based on the detection of a trigger event in the software model.

25. (Original) The method of claim 24, further comprising step:

loading the shared memory with state information from the hardware model.

26. (Previously Presented) The method of claim 25, further comprising step:

simulating a behavior of the circuit using the software model by initially using state information from the hardware model in shared memory.

27. (Original) The method of claim 24, wherein the step of generating the hardware model further comprises steps:

determining component type in the hardware language; and

generating the hardware model based on component type.

28. (Original) The method of claim 24, further comprising steps:

selectively switching to the software model; and

simulating a behavior of the circuit with the software model by providing input data to the software model.

29. (Original) The method of claim 24, wherein the step of evaluating further comprises:

determining the presence of input data to the simulation system;

evaluating clock components;

propagating input data to the hardware model;

detecting the trigger event, wherein the trigger event includes an active clock edge

of the clock components; and

evaluating the input data with the hardware model in response to the active clock edge detection.

30. (Previously Presented) A method of evaluating data in a circuit during a simulation process, comprising: generating a software model of the circuit; generating a hardware model of at least a portion of the circuit; allocating space in shared memory for the software model and the hardware model;

storing in the shared memory a first information of the software model and a second information of the hardware model, where the second information comprises at least one internal state of the hardware model and the software model is capable of directly accessing the second information of the hardware model;

propagating data to the hardware model until the data stabilizes;

detecting a clock edge in the software model; and

evaluating data with the hardware model in response to the clock edge detection in the software model and in synchronization with a hardware-generated clock.

31. (Previously Presented) The method of claim 30, further comprising step:

loading the shared memory with state information from the hardware model.

32. (Original) The method of claim 31, further comprising step:

simulating the circuit with the software model by initially using the state information from the hardware model.

33. (Previously Presented) A simulation system operating in a host computer system for simulating a behavior of a circuit, the host computer system including a central processing unit (CPU), shared memory, and a local bus coupling the CPU to main memory and allowing communication between the CPU and main memory, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising:

a software model of the circuit coupled to the local bus;
software control logic coupled to the software model and a hardware logic element, for controlling the operation of the software model and said hardware logic element;
said hardware logic element coupled to the local bus and including a hardware model of at least a portion of the circuit configured automatically based on component type;

DMA engine for loading state information of the hardware model, where the state information comprises at least one internal state of the hardware model, from the hardware logic element to the shared memory; and

the software model is capable of directly accessing the state information of the hardware model.

34. (Original) The system of claim 33, wherein the software control logic further comprises:

interface logic which is capable of receiving input data and a clock data from an external process, and

clock detection logic for detecting an active edge of the clock data and generating a trigger signal.

35. (Original) The system of claim 34, wherein the hardware logic element further comprises:

clock enable logic for evaluating data in the hardware model in response to the trigger signal.

36. (Original) The system of claim 33, wherein the hardware logic element comprises a field programmable device.

37. (Original) The system of claim 33, wherein the hardware logic element comprises:

a plurality of field programmable devices coupled together, each field programmable device including a portion of the hardware model of the circuit;

a plurality of interconnections to couple the portions of the hardware model

together, each interconnection representing a direct connection between field programmable devices, wherein the shortest path between any two field programmable devices is at most two interconnections.

38. (Previously Presented) A coverification system for verifying a user design, comprising:

- a computing system including a central processing unit and memory for modeling the user design in software;

- an internal bus system coupled to the computing system;

- reconfigurable hardware logic coupled to the internal bus system and for modeling at least a portion of the user design in hardware;

- an external interface coupled to the internal bus system and at least one external device;

- control logic coupled to the internal bus system for controlling the delivery of data among the reconfigurable hardware logic, the computing system, and the external interface; and

- shared memory for loading state information of the user design, where the state information comprises at least one internal state of the user design, from the reconfigurable hardware logic to the computing system.

39. (Previously Presented) The coverification system of claim 38, wherein the control logic further comprises:

- a data-in control logic for controlling delivery of data from the computing system and the external interface to the reconfigurable hardware logic, including

- data-in pointer logic coupled to the internal bus system for generating selective pointer signals to a data-in latch logic, the generation of selective pointer signals based on whether the data is arriving from the computing system or the external interface and the particular internal nodes in the reconfigurable hardware logic selected to be driven, and

- data-in latch logic coupled to the internal bus system, a plurality of internal nodes in the reconfigurable hardware logic, and the data-in pointer logic for delivering data from

the internal bus system to selective internal nodes in the reconfigurable hardware logic in response to the selective pointer signals.

40. (Previously Presented) The coverification system of claim 39, further comprising:

an external buffer coupled to the external interface for storing data originating from the external interface and also coupled to the internal bus system wherein the computing system has access to data in the external buffer.

41. (Original) The coverification system of claim 39, wherein the control logic further comprises:

a data-out control logic for controlling delivery of data from the reconfigurable hardware logic to the computing system and the external interface, including,

data-out pointer logic coupled to the internal bus system for generating selective pointer signals to a data-out gating logic, the generation of selective pointer signals based on whether the data is destined for the computing system or the external interface and the particular internal nodes in the reconfigurable hardware logic selected to be driven, and

data-out gating logic coupled to the internal bus system, a plurality of internal nodes in the reconfigurable hardware logic, and the data-out pointer logic for delivering data from the selective internal nodes in the reconfigurable hardware logic to the internal bus system in response to the selective pointer signals.

42. (Original) The coverification system of claim 38, further comprising:

software clock logic for detecting an active clock edge of a clock signal in the software model during data evaluation, generating a software clock to the reconfigurable hardware logic to control a corresponding data evaluation in the reconfigurable hardware logic, and generating the software clock to the external interface.

43. (Original) The coverification system of claim 38, wherein the computing system further includes at least one model of an external I/O device in software.

44. (Previously Presented) A method of verifying the proper operation of a user design connected to an external I/O device, comprising steps:

- generating a first model of the user design in software;
- loading the first model in a shared memory;
- generating a second model of at least a portion of the user design in hardware;
- controlling the second model in hardware with the first model in the software; and
- loading state information from the second model, where the state information comprises at least one internal state of the second model, to the shared memory and the first model is capable of directly accessing the state information of the second model.

45. (Original) The method of claim 44, wherein the step of controlling further comprises:
synchronizing the data evaluation in the first model in software and the second model in hardware with a software-generated clock.

46. (Previously Presented) The method of claim 45, further comprising steps:
simulating selected debug test points in software;
accelerating selected debug test points in hardware; and
controlling the delivery of data among the first model in software, the second model in hardware, and the external I/O device so that the first model in software has access to all delivered data.

47. (Previously Presented) The method of claim 46, wherein the step of controlling further comprising steps: electing at least one internal node in the reconfigurable hardware logic;
determining if the data being delivered is from the first model in software or the external I/O device; and
generating selected pointer signals to at least one latching logic coupled to the selected internal node based on the selecting and determining steps so that the data is delivered from either the first model in software or the external I/O device to the second model in hardware.

48. (Original) The method of claim 47, further comprising steps:

storing data delivered from the external I/O device in an external buffer coupled to the second model in hardware; and

providing the first model in software access to data in the external buffer.

49. (Original) The method of claim 46, wherein the step of controlling further comprising

steps: selecting at least one internal node in the reconfigurable hardware logic;

determining if the data being delivered is destined for the first model in software or the external I/O device; and

generating selected pointer signals to at least one gating logic coupled to the selected internal node based on the selecting and determining steps so that the data is delivered from the second model in hardware to either the first model in software or to both the first model in software and the external I/O device.

50. (Original) The method of claim 44, further comprising step:

generating a model of an external I/O device in software.

EVIDENCE APPENDIX

[None]

RELATED PROCEEDINGS APPENDIX

[None]